Foundation of Cryptography, Lecture 8
Secure Multiparty Computation

Iftach Haitner, Tel Aviv University

Tel Aviv University.

June 11, 2013

Section 1

The Model

Multiparty Computation

- Multiparty Computation – computing a functionality f

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness
  - Privacy

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness
  - Privacy
  - Independence of inputs

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness
  - Privacy
  - Independence of inputs
  - Guaranteed output delivery

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness
  - Privacy
  - Independence of inputs
  - Guaranteed output delivery
  - Fairness : corrupted parties should get their output iff the honest parties do

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness
  - Privacy
  - Independence of inputs
  - Guaranteed output delivery
  - Fairness : corrupted parties should get their output iff the honest parties do
  - and ...

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a "secure manner"
  - Correctness
  - Privacy
  - Independence of inputs
  - Guaranteed output delivery
  - Fairness : corrupted parties should get their output iff the honest parties do
  - and ...
- Examples: coin-tossing, broadcast, electronic voting, electronic auctions

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner"
  - Correctness
  - Privacy
  - Independence of inputs
  - Guaranteed output delivery
  - Fairness : corrupted parties should get their output iff the honest parties do
  - and ...
- Examples: coin-tossing, broadcast, electronic voting, electronic auctions
- How should we model it?
  Real Vs. Ideal model

Real Model Execution

Let $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ be a pair of algorithms, and $x_1, x_2 \in \{0,1\}^*$. Define $\mathsf{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)$ as the joint outputs of $(\mathsf{A}_1(x_1), \mathsf{A}_2(x_2))$

Real Model Execution

Let $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ be a pair of algorithms, and $x_1, x_2 \in \{0,1\}^*$. Define $\mathsf{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)$ as the joint outputs of $(\mathsf{A}_1(x_1), \mathsf{A}_2(x_2))$

- An honest party follows the prescribed protocol (i.e., π) and outputs the prescribed output

Real Model Execution

Let $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ be a pair of algorithms, and $x_1, x_2 \in \{0,1\}^*$. Define $\mathsf{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)$ as the joint outputs of $(\mathsf{A}_1(x_1), \mathsf{A}_2(x_2))$

- An honest party follows the prescribed protocol (i.e., π) and outputs the prescribed output
- A semi-honest party follows the protocol, but might output additional information

Ideal Model Execution

Ideal Model Execution

Let $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$ be a pair of oracle-aided algorithms. An execution of $\overline{\mathsf{B}}$ in the ideal model on inputs $x_1, x_2 \in \{0,1\}^*$, denoted $\mathrm{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)$, is the joint output of the parties in the end of the following experiment:

Ideal Model Execution

Let $\overline{B} = (B_1, B_2)$ be a pair of oracle-aided algorithms. An execution of $\overline{B}$ in the ideal model on inputs $x_1, x_2 \in \{0,1\}^*$, denoted $\text{IDEAL}_{f,\overline{B}}(x_1, x_2)$, is the joint output of the parties in the end of the following experiment:

1. The input of B_i is x_i ($i \in \{0,1\}$)
2. Each party sends value y_i to the trusted party (possibly $\perp$)
3. Trusted party sends $f_i(y_0, y_1)$ to B_i (sends $\perp$, if $\perp \in \{y_0, y_1\}$)
4. Each party outputs some value

Ideal Model Execution

Let $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$ be a pair of oracle-aided algorithms. An execution of $\overline{\mathsf{B}}$ in the ideal model on inputs $x_1, x_2 \in \{0,1\}^*$, denoted $\mathrm{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)$, is the joint output of the parties in the end of the following experiment:

1. The input of B_i is x_i ($i \in \{0,1\}$)
2. Each party sends value y_i to the trusted party (possibly $\perp$)
3. Trusted party sends $f_i(y_0, y_1)$ to B_i (sends $\perp$, if $\perp \in \{y_0, y_1\}$)
4. Each party outputs some value

- An honest party, sends its input to the trusted party and outputs the trusted party message

Ideal Model Execution

Let $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$ be a pair of oracle-aided algorithms. An execution of $\overline{\mathsf{B}}$ in the ideal model on inputs $x_1, x_2 \in \{0,1\}^*$, denoted $\mathsf{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)$, is the joint output of the parties in the end of the following experiment:

1. The input of B_i is x_i ($i \in \{0,1\}$)
2. Each party sends value y_i to the trusted party (possibly $\perp$)
3. Trusted party sends $f_i(y_0, y_1)$ to B_i (sends $\perp$, if $\perp \in \{y_0, y_1\}$)
4. Each party outputs some value

- An honest party, sends its input to the trusted party and outputs the trusted party message
- A semi-honest party, might output additional information

Securely computing a functionality

$\overline{A} = (A_1, A_2)$ is an admissible algorithm pair for π [resp., for f], if at least one party is honest

Securely computing a functionality

$\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ is an admissible algorithm pair for π [resp., for f], if at least one party is honest

Definition 1 (secure computation)

A protocol π securely computes f, if $\forall$ real model, admissible PPT $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$, exists an ideal-model admissible pair PPT $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$, s.t.

$$\{\mathsf{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)\}_{x_1,x_2} \approx_c \{\mathsf{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)\}_{x_1,x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0,1\}^*$ with $|x_1| = |x_2|$.

Securely computing a functionality

$\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ is an admissible algorithm pair for π [resp., for f], if at least one party is honest

Definition 1 (secure computation)

A protocol π securely computes f, if $\forall$ real model, admissible PPT $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$, exists an ideal-model admissible pair PPT $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$, s.t.

$$\{\mathsf{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)\}_{x_1,x_2} \approx_c \{\mathsf{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)\}_{x_1,x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0,1\}^*$ with $|x_1| = |x_2|$.

- Auxiliary inputs

Securely computing a functionality

$\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ is an admissible algorithm pair for π [resp., for f], if at least one party is honest

Definition 1 (secure computation)

A protocol π securely computes f, if $\forall$ real model, admissible PPT $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$, exists an ideal-model admissible pair PPT $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$, s.t.

$$\{\mathrm{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)\}_{x_1,x_2} \approx_c \{\mathrm{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)\}_{x_1,x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0,1\}^*$ with $|x_1| = |x_2|$.

- Auxiliary inputs
- Security parameter

Securely computing a functionality

$\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$ is an admissible algorithm pair for π [resp., for f], if at least one party is honest

Definition 1 (secure computation)

A protocol π securely computes f, if $\forall$ real model, admissible PPT $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$, exists an ideal-model admissible pair PPT $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$, s.t.

$$\{\mathrm{REAL}_{\overline{\mathsf{A}}}(x_1, x_2)\}_{x_1,x_2} \approx_c \{\mathrm{IDEAL}_{f,\overline{\mathsf{B}}}(x_1, x_2)\}_{x_1,x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0,1\}^*$ with $|x_1| = |x_2|$.

- Auxiliary inputs
- Security parameter
- We focus on semi-honest adversaries

Section 2

Oblivious Transfer

Oblivious Transfer

A protocol that securely realize the functionality
OT: $(\{0,1\}^* \times \{0,1\}^*) \times \{0,1\} \mapsto \{0,1\}^* \times \perp$, where $f_1(\cdot) = \perp$ and
$f_2((\sigma_0, \sigma_1), i) = \sigma_i$ and .

Oblivious Transfer

A protocol that securely realize the functionality
OT: $(\{0,1\}^* \times \{0,1\}^*) \times \{0,1\} \mapsto \{0,1\}^* \times \perp$, where $f_1(\cdot) = \perp$ and $f_2((\sigma_0, \sigma_1), i) = \sigma_i$ and .

- “Complete" for multiparty computation

Oblivious Transfer

A protocol that securely realize the functionality
OT: $(\{0,1\}^* \times \{0,1\}^*) \times \{0,1\} \mapsto \{0,1\}^* \times \perp$, where $f_1(\cdot) = \perp$ and $f_2((\sigma_0, \sigma_1), i) = \sigma_i$ and .

- “Complete" for multiparty computation
- We show how to construct for bit inputs

Oblivious Transfer from Trapdoor Permutations

Let (G, f, Inv) be a TDP and let b be an hardcore predicate for f.

Oblivious Transfer from Trapdoor Permutations

Let (G, f, Inv) be a TDP and let b be an hardcore predicate for f.

Protocol 2 ((S, R))

Common input: 1^n, S**'s input:** $\sigma_0, \sigma_1 \in \{0,1\}$, R**'s input:** $i \in \{0,1\}$

1. S chooses $(e, d) \leftarrow \mathrm{G}(1^n)$, and sends e to R
2. R chooses $x_0, x_1 \leftarrow \{0,1\}^n$, sets $y_i = f_e(x_i)$ and $y_{1-i} = x_{1-i}$, and sends y_0, y_1 to S
3. S sets $c_j = b(\mathrm{Inv}_d(y_j)) \oplus \sigma_j$, for $j \in \{0,1\}$, and sends (c_0, c_1) to R
4. R outputs $c_i \oplus b(x_i)$.

Oblivious Transfer from Trapdoor Permutations

Let (G, f, Inv) be a TDP and let b be an hardcore predicate for f.

Protocol 2 ((S, R))

Common input: 1^n, S**'s input:** $\sigma_0, \sigma_1 \in \{0,1\}$, R**'s input:** $i \in \{0,1\}$

1. S chooses $(e, d) \leftarrow \mathrm{G}(1^n)$, and sends e to R
2. R chooses $x_0, x_1 \leftarrow \{0,1\}^n$, sets $y_i = f_e(x_i)$ and $y_{1-i} = x_{1-i}$, and sends y_0, y_1 to S
3. S sets $c_j = b(\mathrm{Inv}_d(y_j)) \oplus \sigma_j$, for $j \in \{0,1\}$, and sends (c_0, c_1) to R
4. R outputs $c_i \oplus b(x_i)$.

Claim 3

Protocol 2 securely realizes OT (in the semi-honest model).

Proving Claim 3

- Correctness

Proving Claim 3

- Correctness
- Secrecy: We need to prove that $\forall$ real model, semi-honest, admissible PPT $\overline{\mathsf{A}} = (\mathsf{A}_1, \mathsf{A}_2)$, exists an ideal-model, admissible pair PPT $\overline{\mathsf{B}} = (\mathsf{B}_1, \mathsf{B}_2)$ s.t.

$$\{\mathsf{REAL}_{\overline{\mathsf{A}}}(1^n, (\sigma_0, \sigma_1), i\} \approx_c \{\mathsf{IDEAL}_{\mathsf{OT}, \overline{\mathsf{B}}}(1^n, (\sigma_0, \sigma_1), i\}, \tag{1}$$

where the enumeration is over $n \in \mathbb{N}$ and $\sigma_0, \sigma_1, i \in \{0, 1\}$

R's privacy

For $\overline{\mathsf{A}} = (\mathsf{S}', \mathsf{R})$, where S' is a semi-honest implementation of S, let $\overline{\mathsf{B}} = (\mathsf{S}'_{\mathcal{I}}, \mathsf{R}_{\mathcal{I}})$ be an ideal-model protocol, where $\mathsf{R}_{\mathcal{I}}$ acts honestly, and

Algorithm 4 ($\mathsf{S}'_{\mathcal{I}}$)

input: $1^n, \sigma_0, \sigma_1$

1. Send (σ_0, σ_1) to the trusted party
2. Emulate $\mathsf{S}'(1^n, \sigma_0, \sigma_1)$, acting as $\mathsf{R}(1^n, 0)$
3. Output the same output that S' does

R's privacy

For $\overline{\mathsf{A}} = (\mathsf{S}', \mathsf{R})$, where S' is a semi-honest implementation of S, let $\overline{\mathsf{B}} = (\mathsf{S}'_{\mathcal{I}}, \mathsf{R}_{\mathcal{I}})$ be an ideal-model protocol, where $\mathsf{R}_{\mathcal{I}}$ acts honestly, and

Algorithm 4 ($\mathsf{S}'_{\mathcal{I}}$)

input: $1^n, \sigma_0, \sigma_1$

1. Send (σ_0, σ_1) to the trusted party
2. Emulate $\mathsf{S}'(1^n, \sigma_0, \sigma_1)$, acting as $\mathsf{R}(1^n, 0)$
3. Output the same output that S' does

Claim 5

Equation (1) holds with respect to $\overline{\mathsf{A}}$ and $\overline{\mathsf{B}}$.

R's privacy

For $\overline{\mathsf{A}} = (\mathsf{S}', \mathsf{R})$, where S' is a semi-honest implementation of S, let $\overline{\mathsf{B}} = (\mathsf{S}'_{\mathcal{I}}, \mathsf{R}_{\mathcal{I}})$ be an ideal-model protocol, where $\mathsf{R}_{\mathcal{I}}$ acts honestly, and

Algorithm 4 ($\mathsf{S}'_{\mathcal{I}}$)

input: $1^n, \sigma_0, \sigma_1$

1. Send (σ_0, σ_1) to the trusted party
2. Emulate $\mathsf{S}'(1^n, \sigma_0, \sigma_1)$, acting as $\mathsf{R}(1^n, 0)$
3. Output the same output that S' does

Claim 5

Equation (1) holds with respect to $\overline{\mathsf{A}}$ and $\overline{\mathsf{B}}$.

Proof?

S's privacy

For $\overline{\mathsf{A}} = (\mathsf{S}, \mathsf{R}')$, where R' is a semi-honest implementation of R, let $\overline{\mathsf{B}} = (\mathsf{S}_{\mathcal{I}}, \mathsf{R}'_{\mathcal{I}})$ be an ideal-model protocol, where $\mathsf{S}_{\mathcal{I}}$ acts honestly and

Algorithm 6 ($\mathsf{R}'_{\mathcal{I}}$)

input: 1^n,$i \in \{0,1\}$

1. Send i to the trusted party, and let σ be its answer.
2. Emulate $\mathsf{R}'(1^n, i)$, acting as $\mathsf{S}(1^n, \sigma_0, \sigma_1)$, where $\sigma_i = \sigma$, and $\sigma_{1-i} = 0$
3. Output the same output that R' does

S's privacy

For $\overline{A} = (S, R')$, where R' is a semi-honest implementation of R, let $\overline{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be an ideal-model protocol, where $S_{\mathcal{I}}$ acts honestly and

Algorithm 6 ($R'_{\mathcal{I}}$)

input: 1^n,$i \in \{0, 1\}$

1. Send i to the trusted party, and let σ be its answer.
2. Emulate $R'(1^n, i)$, acting as $S(1^n, \sigma_0, \sigma_1)$, where $\sigma_i = \sigma$, and $\sigma_{1-i} = 0$
3. Output the same output that R' does

Claim 7

Equation (1) holds with respect to $\overline{A}$ and $\overline{B}$.

S's privacy

For $\overline{A} = (S, R')$, where R' is a semi-honest implementation of R, let $\overline{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be an ideal-model protocol, where $S_{\mathcal{I}}$ acts honestly and

Algorithm 6 ($R'_{\mathcal{I}}$)

input: 1^n,$i \in \{0, 1\}$

1. Send i to the trusted party, and let σ be its answer.
2. Emulate $R'(1^n, i)$, acting as $S(1^n, \sigma_0, \sigma_1)$, where $\sigma_i = \sigma$, and $\sigma_{1-i} = 0$
3. Output the same output that R' does

Claim 7

Equation (1) holds with respect to $\overline{A}$ and $\overline{B}$.

Proof?

Section 3

Yao Garbled Circuit

Before we start

- Fix a (multiple message) semantically-secure private-key encryption scheme $(\mathsf{G}', \mathsf{E}, \mathsf{D})$ with
  1. $\mathsf{G}'(1^n) = U_n$
  2. $\mathsf{D}_d(\mathsf{E}_{d'}(m)) = \perp$, for any $d \neq d'$ and $m \in \{0,1\}^*$.

Before we start

- Fix a (multiple message) semantically-secure private-key encryption scheme $(\mathsf{G}', \mathsf{E}, \mathsf{D})$ with
  1. $\mathsf{G}'(1^n) = U_n$
  2. $\mathsf{D}_d(\mathsf{E}_{d'}(m)) = \perp$, for any $d \neq d'$ and $m \in \{0,1\}^*$.

  Can we achieve such scheme?

Before we start

- Fix a (multiple message) semantically-secure private-key encryption scheme $(\mathsf{G}', \mathsf{E}, \mathsf{D})$ with
  1. $\mathsf{G}'(1^n) = U_n$
  2. $\mathsf{D}_d(\mathsf{E}_{d'}(m)) = \perp$, for any $d \neq d'$ and $m \in \{0,1\}^*$.

  Can we achieve such scheme?
- Boolean circuits: gates, wires, inputs, outputs, values, computation

The Garbled Circuit

Let C be Boolean a circuit from $\{0,1\}^\ell$ to $\{0,1\}^m$ and let $n \in \mathbb{N}$

The Garbled Circuit

Let C be Boolean a circuit from $\{0,1\}^\ell$ to $\{0,1\}^m$ and let $n \in \mathbb{N}$

- Let $\mathcal{W}$ and G be the (indices) of wires and gates of C.

input wire i	input wire j	output wire h	hidden output wire
k_i^0	k_j^0	$k_h^{g(0,0)}$	$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$
k_i^0	k_j^1	$k_h^{g(1,0)}$	$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$
k_i^1	k_j^0	$k_h^{g(0,1)}$	$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$
k_i^1	k_j^1	$k_h^{g(1,1)}$	$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$

Figure : Table for gate g, with input wires i and j, and output wire h.

The Garbled Circuit

Let C be Boolean a circuit from $\{0,1\}^\ell$ to $\{0,1\}^m$ and let $n \in \mathbb{N}$

- Let $\mathcal{W}$ and G be the (indices) of wires and gates of C.
- For any $w \in \mathcal{W}$, associate two random ‘keys" $k_0^w, k_w^1 \in \{0,1\}^n$.

input wire i	input wire j	output wire h	hidden output wire
k_i^0	k_j^0	$k_h^{g(0,0)}$	$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$
k_i^0	k_j^1	$k_h^{g(1,0)}$	$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$
k_i^1	k_j^0	$k_h^{g(0,1)}$	$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$
k_i^1	k_j^1	$k_h^{g(1,1)}$	$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$

Figure : Table for gate g, with input wires i and j, and output wire h.

The Garbled Circuit

Let C be Boolean a circuit from $\{0,1\}^\ell$ to $\{0,1\}^m$ and let $n \in \mathbb{N}$

- Let $\mathcal{W}$ and G be the (indices) of wires and gates of C.
- For any $w \in \mathcal{W}$, associate two random 'keys" $k_0^w, k_w^1 \in \{0,1\}^n$.
- For $g \in G$ with input wires i, j and output wire h, let $T(g)$ be the following table

input wire i	input wire j	output wire h	hidden output wire
k_i^0	k_j^0	$k_h^{g(0,0)}$	$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$
k_i^0	k_j^1	$k_h^{g(1,0)}$	$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$
k_i^1	k_j^0	$k_h^{g(0,1)}$	$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$
k_i^1	k_j^1	$k_h^{g(1,1)}$	$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$

Figure : Table for gate g, with input wires i and j, and output wire h.

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of C

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of C

- For $g \in G$, let $\widetilde{T}(g)$ be a random permutation of the fourth column of $T(g)$

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of C

- For $g \in G$, let $\widetilde{T}(g)$ be a random permutation of the fourth column of $T(g)$
- For $x \in \{0,1\}^\ell$ and $w \in \mathcal{W}$, let $C(x)_w$ be the value the computation of $C(x)$ assigns to w.

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of C

- For $g \in G$, let $\widetilde{T}(g)$ be a random permutation of the fourth column of $T(g)$
- For $x \in \{0,1\}^\ell$ and $w \in \mathcal{W}$, let $C(x)_w$ be the value the computation of $C(x)$ assigns to w.
- Given $\widetilde{T} = \{(g, \widetilde{T}(g))\}_{g \in G}$, $\{k_i^{C(x)_w}\}_{w \in \mathcal{I}}$, for some $x \in \{0,1\}^\ell$, and $\{(w, k_w)\}_{w \in \mathcal{O}}$, we can efficiently compute $C(x)$

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of C

- For $g \in G$, let $\widetilde{T}(g)$ be a random permutation of the fourth column of $T(g)$
- For $x \in \{0,1\}^\ell$ and $w \in \mathcal{W}$, let $C(x)_w$ be the value the computation of $C(x)$ assigns to w.
- Given $\widetilde{T} = \{(g, \widetilde{T}(g))\}_{g \in G}$, $\{k_i^{C(x)_w}\}_{w \in \mathcal{I}}$, for some $x \in \{0,1\}^\ell$, and $\{(w, k_w)\}_{w \in \mathcal{O}}$, we can efficiently compute $C(x)$
- No other information about x leaks!

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of C

- For $g \in G$, let $\widetilde{T}(g)$ be a random permutation of the fourth column of $T(g)$
- For $x \in \{0,1\}^\ell$ and $w \in \mathcal{W}$, let $C(x)_w$ be the value the computation of $C(x)$ assigns to w.
- Given $\widetilde{T} = \{(g, \widetilde{T}(g))\}_{g \in G}$, $\{k_i^{C(x)_w}\}_{w \in \mathcal{I}}$, for some $x \in \{0,1\}^\ell$, and $\{(w, k_w)\}_{w \in \mathcal{O}}$, we can efficiently compute $C(x)$
- No other information about x leaks!
- Can we use garbled circuit for secure computation?

The protocol

Let $f\colon \{0,1\}^\ell \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^m$ and let C be a circuit that computes f.

The protocol

Let $f\colon \{0,1\}^\ell \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^m$ and let C be a circuit that computes f. Let (S,R) be a secure protocol for OT.

The protocol

Let $f\colon \{0,1\}^\ell \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^m$ and let C be a circuit that computes f. Let (S,R) be a secure protocol for OT.
Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the input wires of x_1 and x_2 (A and B inputs), and let $\mathcal{O}_1$ and $\mathcal{O}_2$ be the output wires of A and B.

The protocol

Let $f\colon \{0,1\}^{\ell} \times \{0,1\}^{\ell} \times \{0,1\}^{m} \times \{0,1\}^{m}$ and let C be a circuit that computes f. Let (S,R) be a secure protocol for OT.
Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the input wires of x_1 and x_2 (A and B inputs), and let $\mathcal{O}_1$ and $\mathcal{O}_2$ be the output wires of A and B.

Protocol 8 ((A,B))

Common input: 1^n. A/B**'s input:** $x_1/x_2 \in \{0,1\}^{\ell}$

1. A prepares random $\{k_w = (k_w^0, k_w^1)\}_{w\in\mathcal{W}}$ and $\widetilde{T}$, and sends $\widetilde{T}$, $\{(w, k_w^{C(x_1,\cdot)_w})\}_{w\in\mathcal{I}_1}$ and $\{(w,k_w)\}_{w\in\mathcal{O}_2}$ to B.
2. $\forall w \in \mathcal{I}_2$, A and B interact in $(\mathsf{S}(k_w), \mathsf{R}(C(\cdot,x_2)_w))(1^n)$.
3. B computes the (garbled) circuit, and sends $\{(w, k_w^{C(x_1,x_2)_w})\}_{w\in\mathcal{O}_2}$ to A.
4. The parties compute $f(x_1,x_2)_1$ and $f(x_1,x_2)_2$ respectively.

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness
2. B's privacy

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness
2. B's privacy

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness
2. B's privacy Immediately follows from the security of the OT

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness
2. B's privacy Immediately follows from the security of the OT
3. A's privacy

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness
2. B's privacy Immediately follows from the security of the OT
3. A's privacy

Claim 9

Protocol 8 securely computes f (in the semi-honest model)

Proof:

1. Correctness
2. B's privacy Immediately follows from the security of the OT
3. A's privacy
   The simulator for B puts random values in $\widetilde{T}$, $\{(w, k_w^{C(x_1,\cdot)_w})\}_{w \in \mathcal{I}_1}$ and $\{(w, k_w^{C(\cdot,x_2)_w})\}_{w \in \mathcal{I}_1}$, and sets $\{(w, k_w)\}_{w \in \mathcal{O}_2}$ according to $f_2(x_1, x_2)$.

Extensions

Extensions

- Efficiently computable f
  Both parties first compute C_f – a circuit that compute f for inputs of the right length

Extensions

- Efficiently computable f
  Both parties first compute C_f – a circuit that compute f for inputs of the right length
- Hiding C?

Extensions

- Efficiently computable f
  Both parties first compute C_f – a circuit that compute f for inputs of the right length
- Hiding C?

Extensions

- Efficiently computable f
  Both parties first compute C_f – a circuit that compute f for inputs of the right length
- Hiding C? All but its size

Malicious model

The parties prove that they act “honestly"

Malicious model

The parties prove that they act “honestly"

1. Forces the parties to chose their random coin properly

Malicious model

The parties prove that they act “honestly"

1. Forces the parties to chose their random coin properly
2. Before each step, the parties prove in $\mathcal{ZK}$ that they followed the prescribed protocol (with respect to the random-coins chosen above)

Malicious model

The parties prove that they act “honestly"

1. Forces the parties to chose their random coin properly
2. Before each step, the parties prove in $\mathcal{ZK}$ that they followed the prescribed protocol (with respect to the random-coins chosen above)

Malicious model

The parties prove that they act “honestly"

1. Forces the parties to chose their random coin properly
2. Before each step, the parties prove in $\mathcal{ZK}$ that they followed the prescribed protocol (with respect to the random-coins chosen above)

More efficient alternatives: “cut and choose"

Course Summary

See diagram

What we did not cover

- “Few" reductions

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions : number theory, lattices

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions : number theory, lattices
- Impossibility results

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions : number theory, lattices
- Impossibility results
- “Real life cryptography" (e.g., Random oracle model)

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions : number theory, lattices
- Impossibility results
- “Real life cryptography" (e.g., Random oracle model)
- Security

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions : number theory, lattices
- Impossibility results
- “Real life cryptography" (e.g., Random oracle model)
- Security
- Differential Privacy

What we did not cover

- “Few" reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions : number theory, lattices
- Impossibility results
- “Real life cryptography" (e.g., Random oracle model)
- Security
- Differential Privacy
- and....